

My Python Experience

DesertPy - 5/22/2018

Elliot Garbus

elliottg2@cox.net

Agenda

Who am I?

Why Python?

Learning Resources

Humbling Lessons from CheckIO

My First Project: Key Learnings

- Mido

- tkinter

- PyInstaller

- Execution from an Icon

Summary

And now
for something
completely different...



Who am I as a developer?

- ASM, FORTRAN, LISP, Modula-2, Pascal, PL/M, C, C++
- Primarily ASM, C and C++ as a professional dev (Embedded/Telecom)
- Lots of 'on the metal' programming and performance optimization.
- Worked closely with C/C++ compiler teams on low-level optimization, and on performance tools

... and all a very long time ago



Why Python?


I knew engineers that were raving fans, but never had the time to explore...

Curiosity: Came across the TIOBE Index, Python #4!

I had some hobby projects in mind!

Free & a good fit

Started Feb 2018

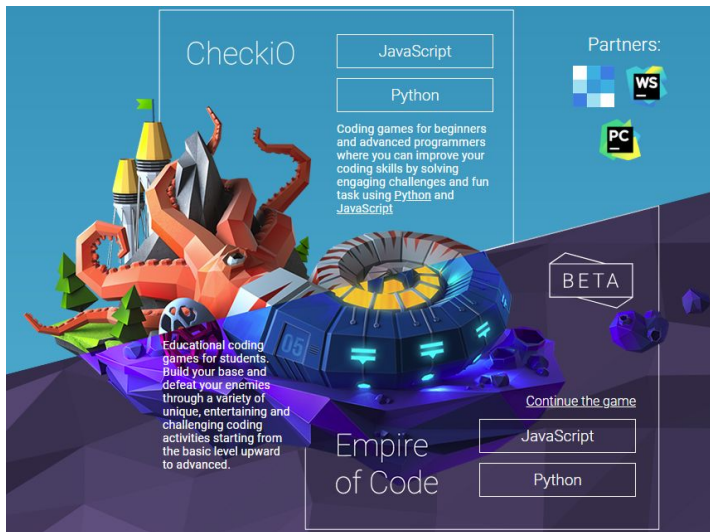
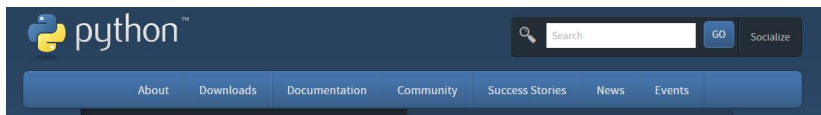
May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%
6	6		Visual Basic .NET	4.124%	+0.73%
7	9	▲	PHP	3.321%	+0.63%
8	7	▼	JavaScript	2.923%	-0.15%
9	-	▲▲	SQL	1.987%	+1.99%
10	11	▲	Ruby	1.182%	-1.25%

My Learning Resources

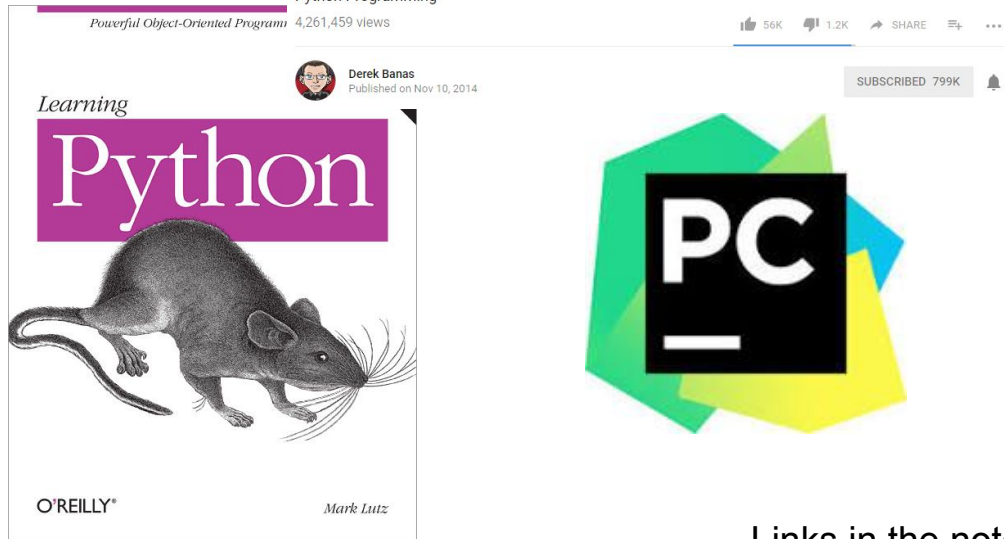


PODCAST.__INIT__

The Podcast About Python and the People Who Make It Great



Python Programming



Links in the notes

Humbling Lessons from CheckIO: Problem 1

Stephan and Sophia forget about security and use simple passwords for everything. Help Nikola develop a password security check module. The password will be considered strong enough if its length is greater than or equal to 10 symbols, it has at least one digit, as well as containing one uppercase letter and one lowercase letter in it. The password contains only ASCII latin letters or digits.

Input: A password as a string.

Output: Is the password safe or not as a boolean or any data type that can be converted and processed as a boolean.

My solution:

```
def checkio(data):  
    digitFound=lowerFound=upperFound=False  
    if len(data)<10:  
        return False  
  
    for c in data:  
        if c.isdigit()==True:  
            digitFound = True  
        if c.islower()==True:  
            lowerFound=True  
        if c.isupper()==True:  
            upperFound=True  
    retcode = digitFound and lowerFound and upperFound  
    return retcode
```



The “Best” Solutions:

```
checkio = lambda s: not(  
    len(s) < 10  
    or s.isdigit()  
    or s.isalpha()  
    or s.islower()  
    or s.isupper()  
)
```

```
import re  
def checkio(data):  
    return True if re.search("(?=.*\d) (?=.*[a-z]) (?=.*[A-Z]).*$", data) \  
        and len(data) >= 10 else False
```



Back to the book!

CheckIO Problem 2:

You are given a text, which contains different english letters and punctuation symbols. You should find the most frequent letter in the text. The letter returned must be in lower case.

While checking for the most wanted letter, casing does not matter, so for the purpose of your search, "A" == "a". Make sure you do not count punctuation symbols, digits and whitespaces, only letters.

If you have two or more letters with the same frequency, then return the letter which comes first in the latin alphabet. For example -- "one" contains "o", "n", "e" only once for each, thus we choose "e".

Best Solution, Another WTF!!!

```
def checkio(text):  
    return max('abcdefghijklmnopqrstuvwxyz', key=text.lower().count)
```

Or:

```
import string ascii_lowercase as letters  
def checkio(text):  
    return max(letters, key=text.lower().count)
```



My Takeaways from CheckIO

Stop thinking in C

Really understand the built-ins & the standard data-type methods

There is a lot of interesting stuff in the standard libraries

Great value is reviewing the code of others

After 4 exercises, I stopped embarrassing myself. Time to move on...



My First Project

Add features to the software that controls a guitar synthesizer

Summary:

Connect over MIDI via USB

Read the current configuration

Draw a UI that reflects the config

Create buttons to program the 4
'Quick knobs'



Choosing a MIDI* library

- Easy to use
- Required features for this and future projects
- Active and Documented
- Selected MIDO - MIDI Objects for Python
 - Library for working with MIDI messages and ports.
 - It's designed to be as straight forward and Pythonic as possible
 - Nice documentation on 'Read the Docs' <https://mido.readthedocs.io/en/latest/#>



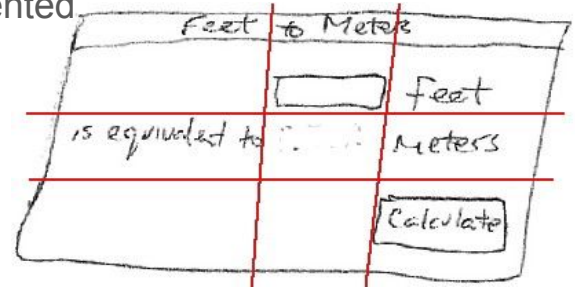
Choosing a GUI framework

- Lots of options, not a lot of guidance...
- Considered:
 - tkinter - a standard python lib, based on TCL/TK
 - WxPython - based on WxWidgets
 - PyQt based of Qt
 - Kivy Python native, focused on games, portability to mobile, multi touch. OGL
- Selected: tkinter
 - This project is a simple UI
 - WxPython was rumored to be poorly documented
 - Qt is big, and dual licensed
 - Project not well aligned to Kivy's strengths



tkinter Lessons Learned

- TCL/TK is an old library that has valued compatibility, there is lots of 'cruft'
 - 3 Geometry Layout schemes: Pack, Grid, Placer
 - Use Grid, use TTK Widgets
- TKDocs is Awesome: <http://www.tkdocs.com>
- It is quick and easy to build a GUI with tkinter
 - There are a few important elements that are not well documented
- GUI layout tool, not required... Avoid PAGE
- Fun, interactive rapid development



“Developers first wanting to learn Tk can be overwhelmed at the range of documentation out there, much of it incredibly out of date”

tkinter: Adding an Icon Image

```
class QQKGUI:
    # The GUI
    def __init__(self, master):
        self.master = master
        self.master.title("QQK")
        self.master.wm_attributes('-topmost', True) # Window always on top
        self.master.wm_geometry('+7-100') # 7 pixels from the left, 100 from the bottom
        img = PhotoImage(file='knob.png')
        self.master.wm_iconphoto(self.master, img)
```

...

- Not well documented:
 - Adding an Icon image
 - Setting “-topmost”
 - `root.wm_*` feels ‘incomplete’



Finished!





PyInstaller: Creating a '.exe' to share

- Freezes (packages) Python programs into stand-alone executables
 - Windows, Linux, Mac OS X, FreeBSD, Solaris and AIX.
 - Works with Python 2.7 and 3.3—3.6
- Very easy to use, very good documentation
- On my Windows system, loads much slower than a unprocessed Python program



PyInstaller
#!/bin/env python
import os
import sys
#####000110001000110001100
#####1101001000111011010
#####1100011100001100011

PyInstaller: Lessons Learned

Needed to expose 'Hidden imports'

```
import mido
import mido.backends.rtmidi # required for pyinstaller to create an exe, in the mido docs
```

Need to inform Pyinstaller about associated files, and where to put them in the 'frozen package'. Use the .spec file to make it easy to repeat builds

```
datas=[('knob.png', '.')]
```

Add code to detect if the program was executing frozen or not, and grabbing the file from the correct place.

```
if getattr(sys, 'frozen', False): # required so iconfile can be packed by pyinstaller
    application_path = sys._MEIPASS
elif __file__:
    application_path = os.path.dirname(__file__)
iconfile = 'knob.png'
img = PhotoImage(file=os.path.join(application_path, iconfile))
```



PyInstaller
#!/bin/env python
import os
import sys
#####0000110001100001100
#####01100100011011010
#####00001100001100011

qqknob.spec

```
# -*- mode: python -*-

block_cipher = None

a = Analysis(['qqknob.py'],
             pathex=['C:\\Users\\Elliot and Sharon\\PycharmProjects\\QQKnob\\venv'],
             binaries=[],
             datas=[('knob.png', '.')],
             hiddenimports=[],
             hookspath=[],
             runtime hooks=[],
             excludes=[],
             win no prefer redirects=False,
             win private assemblies=False,
             cipher=block_cipher)
pyz = PYZ(a.pure, a.zipped_data,
          cipher=block_cipher)
```

```
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          name='qqknob',
          debug=False,
          strip=False,
          upx=True,
          runtime tmpdir=None,
          console=False , icon='knob.ico')
```



PyInstaller

#! /bin/sh
import os
import sys

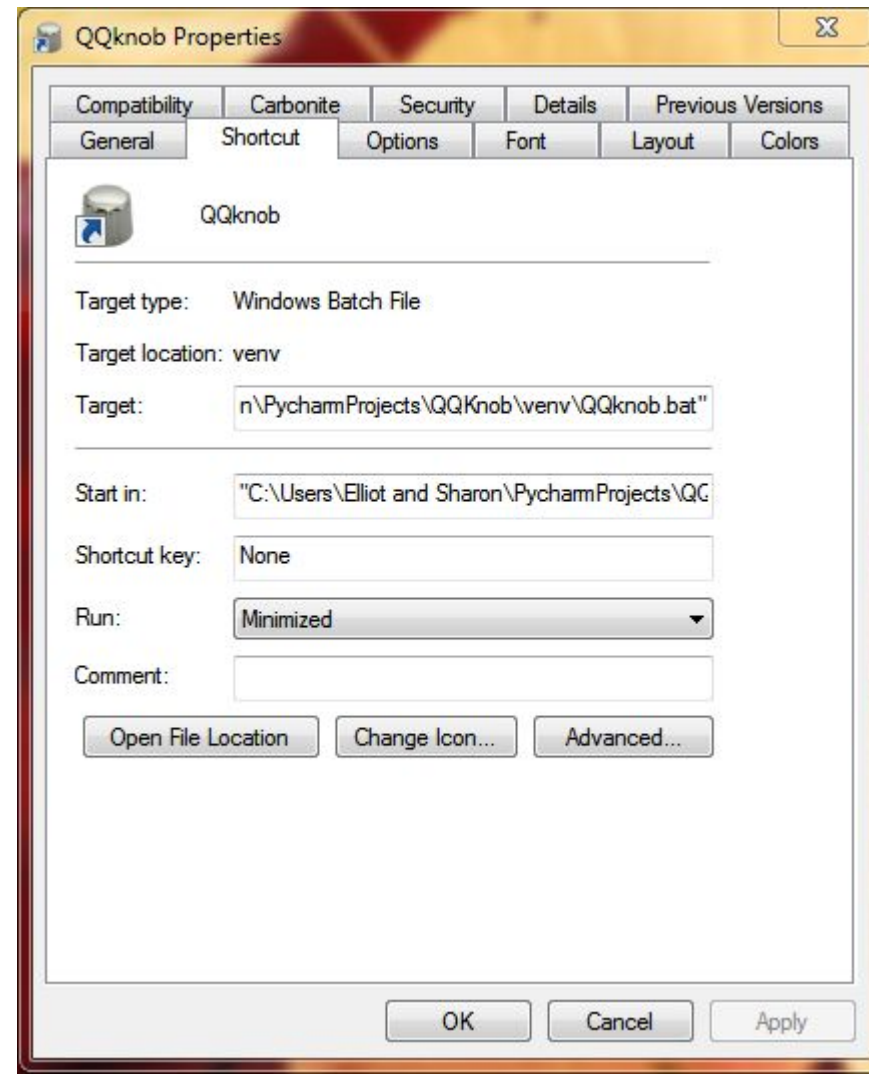
#####110001100010001110001100
#####11010010001110111010
#####11000111000011100011



Creating an Icon to run Python
code from the Desktop

Running from the Desktop

- Create a batch file in your venv folder
 - Right click and drag to the desktop to create a shortcut
 - Right click the icon, select properties, and change the icon
- Note: Use online tools to convert a JPG or PNG to a windows .ico file
- Select Run: 'Minimized', to NOT open a command window



Summary

- Write Pythonic Code
 - Know and use the data structures: string, list, dictionary...
 - Know the builtins & the libs
 - Comprehend comprehensions
 - Learn to love 'in'
 - Avoid your "old" language idioms
- Enjoy the journey
- Share at the meetup

